Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 4 (2011) 2206–2215

Procedia Computer Science

International Conference on Computational Science, ICCS 2011

Distributed Contract Negotiation System for Virtual Organizations

Marcin Stelmach[a], Bartosz Kryza[a,b,*], Renata Slota[a], Jacek Kitowski[a,b]

[a]*Institute of Computer Science AGH-UST, Cracow, Poland*
[b]*Academic Computer Centre CYFRONET-AGH, Cracow, Poland*
{stelmach, bkryza, rena, kito}@agh.edu.pl

Abstract

Nowadays, with the broadening usage of distributed systems and Grid, the need for cooperation between many different heterogeneous organizations occurs. Such cooperation usually requires sharing of access to data, service and other tangible or intangible resources. To accomplish such cooperative efforts between organizations, an idea of the Virtual Organization (VO) emerged. It allows the organizations to share resources between the partners, while enforcing proper quality of service and security. In order to enable creation of such Virtual Organization, it is necessary to allow the participants of the VO to negotiation an agreement concerning the rules of cooperation, which will be encoded in a machine processable format.

In this paper we present the application of the ESB integration architecture in the specialized distributed contract negotiation service - the FiVO DCNS, which is responsible for distributing contract negotiation events and queries between components of FiVO deployed in different organizations.

Keywords: Virtual Organization, Virtual Research Communities, SOA, contract negotiations, ontology, ESB

1. Introduction

Modern applications of Service Oriented Architectures or Grid computing are oriented to allow distinct heterogeneous organizations to share their resources in order to pursue some goal through advanced collaboration schemes supported by their IT infrastructures. The Grid idea introduced the concept of the Virtual Organization, which abstracts the notion of organization into a virtual environment based on distributed computing infrastructures of organizations which want to collaborate. The idea of VO allows these partners to define rules of cooperation in terms of authorization policies or SLA parameters. The process of sharing resources and services defines a model for building large scale computing or information systems. Recently, such systems are often implemented using the Service Oriented Architecture (SOA) paradigm.

In this paper we present Distributed Contract Negotiations System (DCNS) which is a part of the Framework for Intelligent Virtual Organization (FiVO) [1] and is responsible for distributing contract negotiation process events and communication between other components. The main feature of DCNS is to support contract negotiations and management component, which enables coordinated establishment of agreement among partners who want to create

*Corresponding author
Email address: bkryza@agh.edu.pl (Bartosz Kryza)

Selection and/or peer-review under responsibility of Prof. Mitsuhisa Sato and Prof. Satoshi Matsuoka
doi:10.1016/j.procs.2011.04.241

a new Virtual Organization. Many problems related to ad-hoc creation of the VO are mostly related to heterogeneity of resources shared by VO members. All data formats, service descriptions, knowledge repositories can be different. These issues require a system which makes collaboration between the partners possible and efficient. The paper is organized as follows. In section 2 we describe FiVO framework and negotiation process. Section 3 presents a concept of the DCNS architecture and section 4 an application of the ESB architecture to the system implementation. In Section 5 related work in the area of using architectures for Virtual Organization creation is presented. Finally, Section 6 presents conclusion and plans for future work.

2. FiVO framework and contract negotiation process

The FiVO framework is based on the Service Oriented Architecture (SOA) technology, in order to maximize its interoperability with most modern business integration solutions as well as proprietary business solutions The backbone of the system is the Enterprise Service Bus (ESB) technology, which provides communication layer between all the system components. All information is stored in a distributed semantic knowledge base (Grid Organizational Memory - GOM), which provides SOA compatible interface for management and evolution of knowledge in the Web Ontology Language (OWL) format. The contract itself is defined using the OWL ontology, which provides set of concepts and relations describing different types of statements supported by our solution. The negotiation process is supported by an Eclipse based user interface and DCNS instances responsible for distributing and synchronizing the negotiation messages between the partners of the negotiation process, which is additionally supported by a natural language where individual comments or statements can be made using textual form automatically converted to OWL statements using base ontology, obtained from the generic as well as domain specific ontologies of the particular organizations. The FiVO framework overall architecture is presented in Fig. 1.

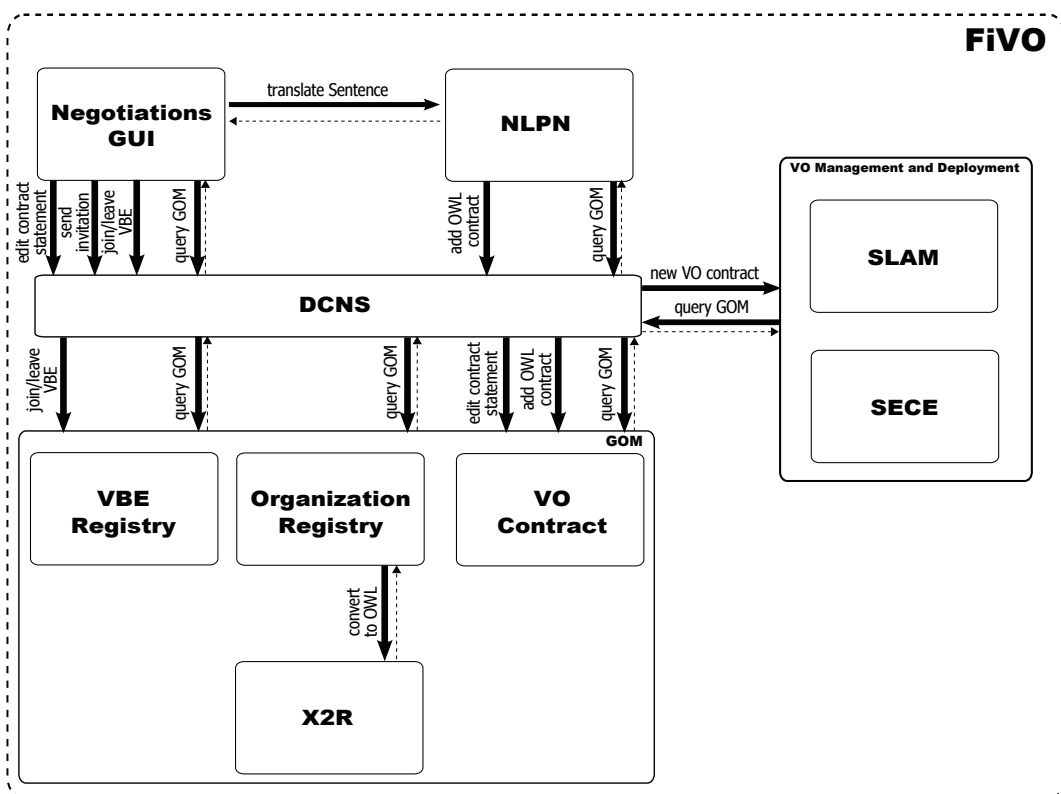

Figure 1: Components of the FiVO.

The GOM knowledge base [2] stores all the necessary information in a unified semantic form. This includes several semantic registries storing information about the organizations assets, resources and competencies, as well as

registries for particular VOs, where information related to contract and its fulfillment is tracked. Since the requirement that organizations provide a semantic description of their resources is often too restrictive, a special tool was developed, called X2R [3]. This tool allows automatic translation of the information about the resources and capabilities they possess, stored in legacy data sources including relational databases, LDAP directories and XML files, and synchronizing this information with the proper GOM instances in the infrastructure. The information stored in GOM can later be used during the negotiation process to present the relevant resources of each organization to the negotiators, which can be then directly used to create contract statements. Once the contract is negotiated, it is stored in GOM and can later be used by VO Deployment components to automatically create the Virtual Organization, i.e. to configure a needed middleware infrastructure within the organizations. The Negotiation Service ensures that the contract statements exchanged between the partners are sent to the proper FiVO instances in other organizations, as well as intermediates in the communication between FiVO components and the GOM knowledge base.

2.1. Contract negotiation process

The contract negotiation process which is supported by DCNS is presented in Figure 2.

During the negotiation phase 2 roles are distinguished from the partners participating in the negotiations. First one, the VO Administrator is the lead participant who can define the VO goal, invite new partners, start the negotiation process and conclude the negotiations and deploy the new VO. The second role is the VO Participant, a regular partner invited by the VO Administrator who can create and modify the contract statements, as well as accept or reject statements regarding their organization.

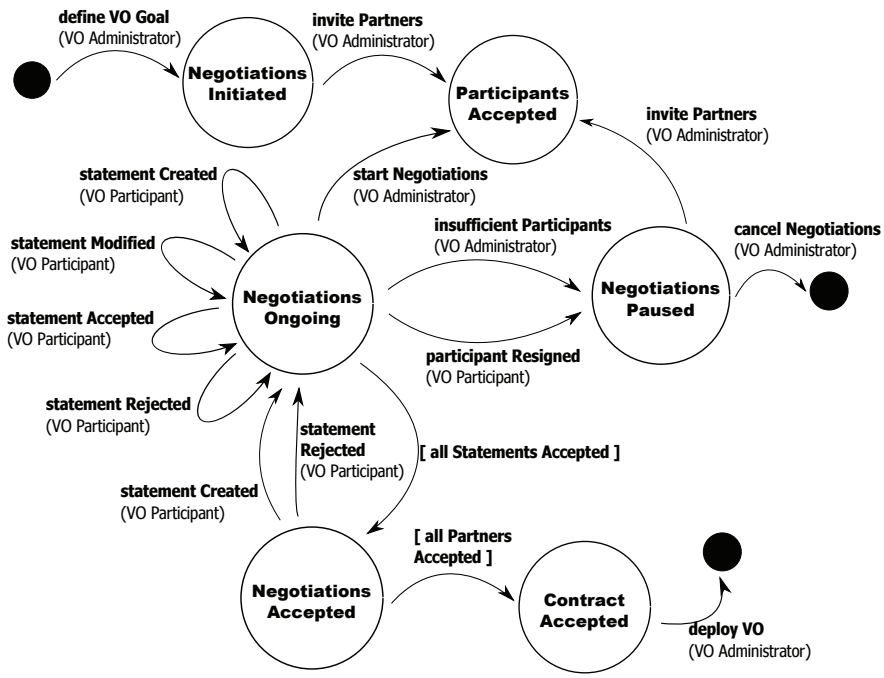

Figure 2: Contract negotiation process states

VO Administrator starts the negotiation process by defining the overall VO goal. This has the form of a set of contract statements (e.g. "*Organization A will provide Service X*"), which define the initial contract. Next, the VO Administrator sends the invitations to the organizations within the VBE (Virtual Organization Breeding Environment) in which the new VO should be created, which are propagated to appropriate organizations via DCNS component. After all partners accept or reject the invitation, the VO Administrator can start the negotiation process. In this phase organizations representatives can take the role of VO Participant, and can create, modify, accept and reject statements in the negotiation tables. Negotiation tables provide means for grouping negotiation on sets of related statements between subgroups of participants. When all statements in a particular table are accepted the table can be accepted

and then becomes part of the contract. When all tables are accepted, i.e. all statements are accepted and there are no opened tables, the negotiations are in Accepted state. In this state, and only in this state, the VO Administrator can ask all partners to accept the contract as whole, which finally constitutes the new Virtual Organization.

3. Distributed Contract Negotiations Service

Design and implementation of a system which allows for distribtued contract negotiation and automatically deploys the VO in the IT infrastructures of the participating organizations is a challenge. Our system had to accomplish the following requirements:

- Support for dynamic, semi-automatic creation and deployment of Virtual Organizations aiming to pursue some goal,
- Fully decentralized nature of the system,
- Architecture and implementation according to the SOA paradigm,
- Support for semantic description of organization resources and capabilities,
- Full support for semantic description of event negotiations, and contracts [4],
- Executing additional services according to the contract statement,
- Handling multiple VO and contracts,
- High scalability and robustness: the process of joining new organization to the system should be straightforward,
- Very simple and lightweight deployment process,

As we mentioned in the introduction, The Distributed Contract Negotiations Service was implemented as a part of the FiVO framework (Framework for intelligent Virtual Organization). The VO creation and management process is based on the negotiated contracts [5], which are semantically defined. Distributed Contracts Negotiation System is a set of activities which enables distributed contract negotiation via exchanging suitable (informations) announcements between organizations. Four kinds of services, which are handled by adequate queues for communication between components:

- services responsible for storing and managing organization description registries,
- services responsible for handling Virtual Organiazation breeding environment description registries,
- services responsible for distributing contract negotiations events - VO Contract Ontology [5][6],
- services responsible for automatically executing services required by VO within IT infrastructures of organizations.

3.1. Motivation of the choice of ESB architecture

The Enterprise Service Bus (ESB) [7] is a software architecture which provides fundamental services for complex architectures via an event-driven and standards-based messaging engine (the bus). An ESB generally provides an abstraction layer on top of an implementation of an enterprise messaging system, which allows integration architects to exploit the value of messaging without writing code. An ESB does not itself implement a service-oriented architecture (SOA) [8] but provides the features with which one may implement such. Any standard JBI or OSGi-compliant service engine or binding component including BPEL, XSLT or JMX engines may be deployed to ESB bus. Developers typically implement an ESB using technologies found in a category of middleware infrastructure products, usually based on recognized standards. This integration enables possibility of cooperation between different kinds of companies and institutions. Consequently it suits perfectly to the concept of the designed system, because just in this system of geographically distributed different services and resources is needed. While integration is inherently complex, one of the major problem is the lack of a common vocabulary and body of knowledge around asynchronous messaging architectures used to build integration solutions. Enterprise integration patterns provide a standard notation for describing how to integrate services. They include several design patterns and are described in [9]. A reliable, asynchronous communication among the services which is provided by the EIP implementations like Apache Camel [10] as well as the messages transformations and routing patterns are very useful in many services based system.

The use of the ESB architecture in case of the FiVO DCNS gives other benefits, like:

- Process of deployment is very simple and lightweight - connecting new binding component to the service bus is quick and requires only providing a message endpoint,
- High reliability - the Fuse ESB implementation uses the ActiveMQ [11] technology as a message broker which supplies us with reliable communication between services with message persistence (in case of system failure),
- High robustness - in the Fuse ESB implementation we can update and connect new services during runtime, without necessity of destroying any existing services.

3.2. Architecture of the FiVO DCNS

In case of the Distributed Contract Negotiations System development, Fuse ESB 4 implementation of the ESB architecture is used [12]. From the five components of the system which were presented in Fig. 3 two are responsible for communication with external systems (Client and Reaction Service), whilst the other components have only the internal ESB endpoints.

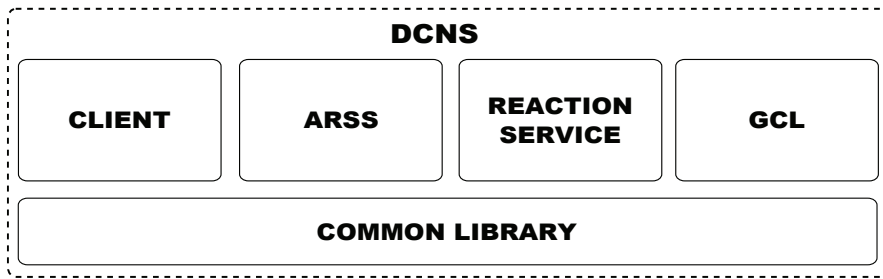

Figure 3: Components of the FiVO DCNS.

The goal of the DCNS is to provide the cooperation between other FiVO components with the accomplishment of all the requirements presented before. In the final design of this system five different services were extracted. Fig. 3 presents the DCNS components model.

The **ARSS (Automatically Run Service Subsystem)** is responsible for deploying application services within the IT infrastrctures of organizations participating in a VO, in response to messages from Reaction Service. This service is responsible for executing services required by the VO, either during creation of the VO, or in case some SLA parameters is not fullfilled, and execution of additional service could improve that aspect of VO execution. All these rules can be expressed in the contract statements using ontological conceptes.

The **Reaction Service** is responsible for reasoning on the contract received from DCNS Client and publishing informations about services which don't fulfill the contract statement and which are to be run for suitable organizations.

The **GCL (Gui Common Library)** is a common library for the GUI and the DCNS. GCL is a set of objects providing a serialization and deserialization library for ontological concepts and other objects that are exchanged between negotiation interfaces and knowledge base components. Instead of XML or OWL, serialized objects of this library are sent between GUI and DCNS Client, thus the solution is not limited to particular semantic technology.

The **Common Library** is a set of common classes for all DCNS components (e.g. queues names, configuration classes, etc.).

The **DCNS Client** is the main component of Distribution Contract Negotiations System, responsible for communication, querying and reasoning over ontologies stored in the GOM instances. DCNS Client can be seen as a proxy between the GOM instances and the rest of the FiVO framework. The architecture of this component is presented in Fig. 4.

DCNS Client has 3 main layers:

- **Communication layer** responsible for communication between other components and GOM knowledge base [2]. Communication between FiVO components proceed via Apache ActiveMQ (JMS messages) and the connection between DCNS Client and GOM goes through Web Services (Apache CXF). The reason to use Web Services for communication is that JMS communication was not implemented in GOM.

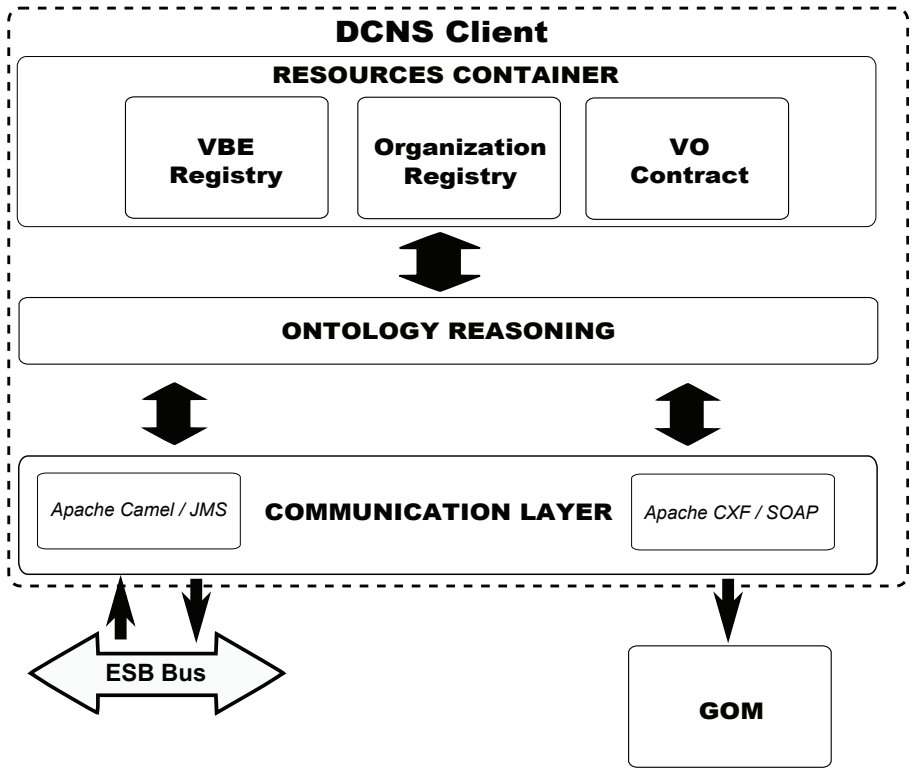

Figure 4: Overview of the FiVO DCNS Client architecture.

- **Ontolgy reasoning layer** responsible for adding, updating, deleting and reasoning over on ontologies and supporting the FiVO graphical user interface for resource browsing, editing and contract negotiation processes, as well as other components (e.g. SLAM [13]).
- **Resources container layer** contains CNO, VBE VO Contract ontology resources. Was created to enable fast reasoning and downloading of data for GUI. Resource layer can be also used as a cache - if GOM is not available in same moment, DCNS Client contener keeps temporarily actual resources and updates them when GOM becomes available.

The DCNS Client supports a several messages types:

- adding, deleting and modifying organization resources,
- invitating participant to VO,
- initializing contract and creating VO for it,
- creating and deleting negotiation tables for contracts,
- adding, deleting and modifying contract statements,
- accepting and rejecting contracts, negotiation tables and contract statements by Participant.

3.3. Ontology abstraction layer and DCNS message format

Since all information related to both the definition of resources as well as the negotiation process and the resulting contract is stored in the OWL format, it became necessary to develop an abstraction library allowing convenient serialization of messagess between the GUI instances and the underlying GOM knowledge base through DCNS component. GOM supports 2 update mechanism: adding entire sets of triples to the ontology or performing fine grained modification on the ontologies using custom OWL change ontology [2]. On the other hand, any changes made in the GOM ontologies by other means than GUI (e.g. using the mentioned X2R library), needed to be updated in the GUI.

This was achieved through implementation of an abstraction library in the form of a set of Java classes, representing all major aspects of OWL (including classes, individuals, properties etc.) with focus on serialization of events generated at the GUI level during the negotiation process.

An example of message modifying a value of property representing particular resource is presented below:

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:j.0="http://fivo.cyf-kr.edu.pl/ont/ChangeOntology#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

        <owl:Ontology rdf:about="http://fivo.cyf-kr.edu.pl/ont/ChangeOntology">
          <owl:imports rdf:resource="http://fivo.cyf-kr.edu.pl/ont/ChangeOntology"/>
        </owl:Ontology>
        <owl:Ontology rdf:about="http://local/change#"/>
                <j.0:IndividualPropertyValueAddition rdf:about="http://local/change#change1">
          <j.0:newValue>"Product 4"@en</j.0:newValue>
          <j.0:target>http://kt.ijs.si/software/CNOntology.owl#Product-Service.name</j.0:target>
          <j.0:affectedResource>http://fivo.cyf-kr.edu.pl:1690/gom/ont/ChatHotels/CNO#prod4
          </j.0:affectedResource>
        </j.0:IndividualPropertyValueAddition>
</rdf:RDF>
```

3.4. Deployment of FiVO DCNS

DCNS was designed in the way that every organization hosts it's own instances of DCNS Client and ARSS Service which is responsible for running services only for this organization. Reaction Service has to be run in at least one of organizations participating in the VO. This configuration is shown in Fig. 5.

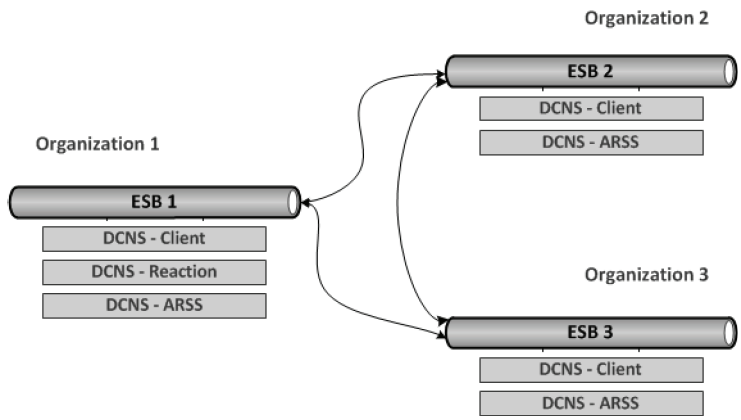

Figure 5: Distributed deployment model of the FiVO DCNS.

The efficiency of this model is mostly related to the ontological reasoner used in DCNS Client, currently 2 reasoners are available: Jena [14] and Pellet [15]. From the conducted tests we can conclude, that the communication between the ESB instances which is performed by the ActiveMQ technology is quite robust and reliable. The most important thing is the proper configuration of the ActiveMQ brokers.

As it was mentioned early the Fuse ESB implementation uses ActiveMQ technology as a message broker. But, more importantly, the ESB architecture allows to introduce special transformations of messages which are defined as the EIP. A full list of these patterns can be found in [9]. Fig. 6 presents the communication between particular DCNS components. Diagram shows the situation when additional application is run as a result of receiving the adequate request from monitoring layer. Reaction Service receives from SLAM subsystem the notification to run the service, sents a request about a contract to DCNS Client (Request-Reply Pattern) and after receiving the contract, decides which service for which organization should be run. Data coming from adequate service are being send to ARSS subsytem instance, which runs proper service.

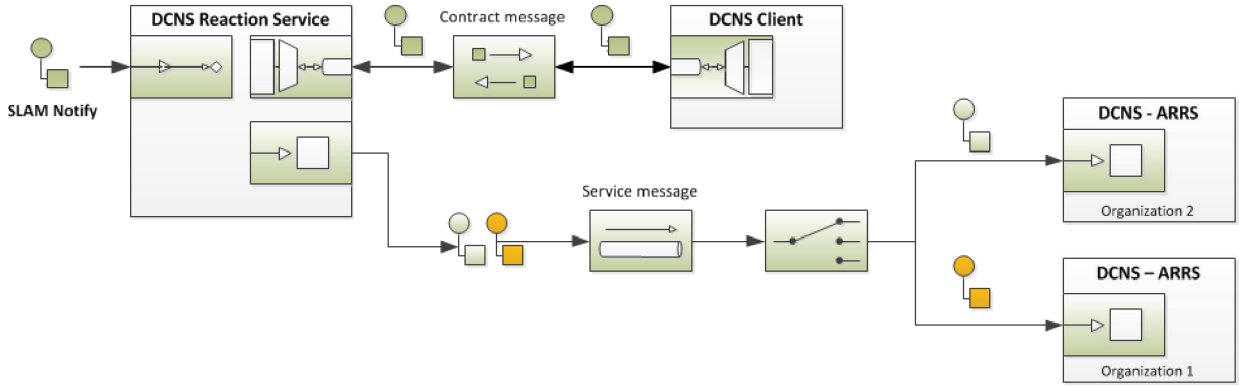

Figure 6: EIP diagram for DCNS components showing the reaction for the failure of the contract

4. Applications of DCNS

In the FiVO environment DCNS fulfills several major roles. The most important one is to support the graphical user interface for Virtual Organization creation support. Contract negotiations are being supported not only by DCNS but also by NLPN [16], which was added to GUI (see Fig. 7).

GUI collects all data from the DCNS. Communication between DCNS Client and GUI proceed via Apache ActiveMQ (JMS messages) DCNS Client processes every request for resources and then returns the data via Request-Reply Pattern. Moreover, contract negotiations and the creation of VO is done using the DCNS Client using other EIP patterns. Requests from GUI are being sent directly to the DCNS Client and which stores the data of the Organization, VBE or Contract. Detention of resources has been described in the preceding paragraph.

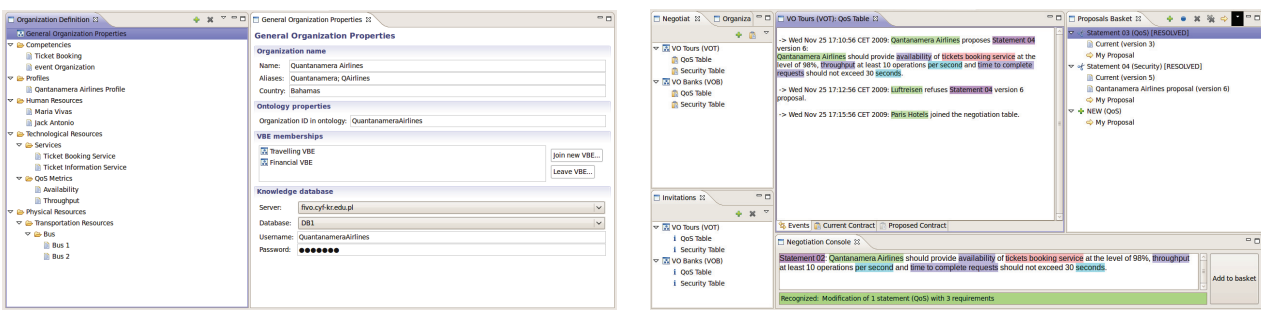

Figure 7: View of organization definition and negotiation perspectives in the FiVO GUI

Another important role is to work with monitoring services (SLAM, SemMon2). SemMon2 [17] is an extensible monitoring platform which implements a semantic-oriented approach to the application monitoring process. SemMon2 provides a semantic-based integration layer for so called physical or low level monitoring systems (e.g. JMX [18] or J-OCM [19]). DCNS is responsible for reactions of the failure of the contract (i.e. a launch services for the Organization). During the negotiations of the contract, participant can define a penalty clause for each statement of the contract. A simple example of the penalty clause can be a launching service. Adding a new organization is very simple, we only deploy an additional ESB bus and all DNCS components for the Organization. Of course, we must note that the DCNS cannot fulfill its tasks without the GUI environment and other FiVO components. This set of functionalities opens FiVO to applications ranging from proprietary business deployments, through research and scientific Grid infrastructures to public and open source integration of portals and social networks.

In particular the application of the negotiation process is analysed currently from the perspective of implementing the idea of Virtual Research Communities (VRC) in the PL-Grid project [20]. The applicability of FiVO is strengthened by its support for Virtual Breeding Environment management (the equivalent of VRC), within which dynamic

Virtual Organizations can be created by the members of such VRC, provided they have means to discover the competencies of each partner and to create an agreement about the future cooperation which is encoded in a machine processable format, which can be used to automatically deploy such VO within the Grid infrastructure.

5. Related work

Created dynamically, adaptable to the needs of virtual organizations, grid environments must be characterized by a high scalability and must support the integration of different areas of research and applications. This poses a certain problem in heterogeneous environments such as Grid, where several organizations need to agree on common rules of resource sharing. When designing the system we considered a number of existing approaches to create the Virtual Organization based on the Model-Centric Development, Agent Systems or SOA architecture.

The Model-Centric Development is based on a model-centric architecture, such as model-driven architecture, defined as well managed web of well-connected models would allow capturing various aspects of information systems engineering, which would align with the tendency to separate business knowledge expression from implementation, that would preserve invariant of business domain out from the technology evolution. Moreover, the use of metamodeling is demanded to bridge similar facets representing similar concerns, yet described by various models using different semantics. A similar work is ATHENA Interoperability Framework described in [21], consisting of a multilayer model for the enterprise, describing business, process, services and information/data layers. To guarantee interoperability, the framework makes use of models for formalizing and exchanging artifacts needed to enable conceptual and technical integration.

Another approach is based on Agent Systems. The usage of software agents to support collective execution of business processes is included in the work of [22] where software agent implementations are generated out from PSM process models, and the work of [23], where software agents run business processes represented in the form of goal-subgoal hierarchies with associated agent plans. An example of the usage of a system of agents to support the formation and resilient operation of Virtual Organizations is CONOISE-G [24]. The usage of software agents for representing enterprises during contract establishment and further execution is described in the work of [25].

The driving goal of the SOA architecture is to integrate together different types of services and resources which belong to different companies [8]. This integration enables possibility of cooperation between different kinds of companies and institutions. Consequently it suits perfectly to the concept of the designed system, because just in this system of geographically distributed different services and resources is needed (see section 3 and 3.1 for further details). Our solution, DCNS allows for dynamic and distributed inception of a Virtual Organization based on a collaborative negotiation between participating organizations with used semantic description [5][6]. The main innovation of the DCNS includes distributed, semantic based contract negotiation system allowing parties to agree on the rules of cooperation within a networked enterprises, automatic and dynamic creation of the Virtual Organization based on the contract by proper configuration of the underlying legacy middleware and architecture and implementation according state of the art SOA technologies.

6. Conclusions and future work

In this paper we presented the Distributed Contract Negotiation System (DCNS). We described architecture and implementation details which would be useful for the usage of the ESB architecture in other projects. An important feature of DCNS is its scalability. Adding new partners to negotiate is flexible and versatile, because if we will add new organization to negotiation, we only need to run new DCNS client. In this paper we didn't present performance and quantitative tests, because the efficiency of network brokers is abundantly sufficient. One broker is able to easily handle 2,000 messages per second. This amount is fully sufficient for our solutions. An important factor in these kind of problems is the reliability, which in our case is 100%. DCNS was created to connect all components of the FiVO Suite (ex. SLAM, GOM, SECE, NLPN). DCNS is responsible for fully automatic integration of all components of the FiVO Suite. In our view, a system designed by us, can be used in other contract negotiations systems based on ESB architecture because the DCNS communication interfaces are universal. The important thing one need to emphasize is that due to DCNS usage, FIVO framework is fully adaptable. We think that these systems can be used in future Organisations which wish to begin to cooperate with each other.

Acknowledgment. This research is partially supported by European Regional Development Fund programs PL-Grid POIG.02.03.00-00-007/08-00 and ITSOA POIG.01.03.01-00-008/08-01.

References

[1] K. Skalkowski, J. Sendor, M. Pastuszko, B. Puzon, J. Fibinger, D. Krol, W. Funika, B. Kryza, R. Slota, J. Kitowski, SOA-Based Support for Dynamic Creation and Monitoring of Virtual Organization, in: S. Ambroszkiewicz, J. Brzezinski, W. Cellary, A. Grzech, K. Zielinski (Eds.), SOA Infrastructure Tools Concepts and Methods, Poznan University of Economices Press, Poznan 2010, pp. 345–374.

[2] B. Kryza, J. Pieczykolan, J. Kitowski, Grid Organizational Memory: A Versatile Solution for Ontology Management in the Grid., in: e-Science, IEEE Computer Society, 2006, p. 16.

[3] X2R Project website.
URL `http://fivo.cyf-kr.edu.pl/trac/fivo/wiki/X2R`

[4] B. Kryza, Ł. Dutka, R. Słota, J. Kitowski, Dynamic VO Establishment in Distributed Heterogeneous Business Environments, in: ICCS 2009: Proceedings of the 9th International Conference on Computational Science, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 709–718.

[5] M. Zuzek, M. Talik, T. Świerczyński, C. Wiśniewski, B. Kryza, Ł. Dutka, J. Kitowski, Formal Model for Contract Negotiation in Knowledge-based Virtual Organizations, in: ICCS '08: Proceedings of the 8th international conference on Computational Science, Part III, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 409–418.

[6] J. Zieba, B. Kryza, R. Slota, L. Dutka, J. Kitowski, Ontology Alignment for Contract Based Virtual Organizations Negotiation and Operation, in: PPAM, Vol. 4967 of Lecture Notes in Computer Science, Springer, 2007, pp. 835–842.

[7] D. A. Chappell, Enterprise Service Bus, O'Reilly Media, Inc., 2004, pp. 1–56, 101–145, 225–232.

[8] M. P. Papazoglou, W.-J. Heuvel, Service oriented architectures: approaches, technologies and research issues, The VLDB Journal 16 (3) (2007) 389–415.

[9] G. Hohpe, B. Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, Pearson Education, Inc., 2004, pp. 99–137, 225–355.

[10] The Apache Camel, http://camel.apache.org/.

[11] The apache software foundation ActiveMQ, http://activemq.apache.org.

[12] The Fuse Open Source Community download site, http://fusesource.com/downloads/.

[13] K. Skalkowski, J. Sendor, R. Slota, J. Kitowski, Application of the ESB Architecture for Distributed Monitoring of the SLA Requirements., in: ISPDC, IEEE Computer Society, 2010, pp. 203–210.

[14] B. McBride, Jena: a semantic web toolkit, IEEE Internet Computing 6 (6) (2002) 55–59.

[15] E. Sirin, B. Parsia, B. Grau, A. Kalyanpur, Y. Katz, Pellet: A practical OWL-DL reasoner, Journal of Web Semantics 5 (2) (2007) 51–53.

[16] M. Pastuszko, B. Kryza, R. Slota, J. Kitowski, Natural Language based Processing of Multilingual Contracts for Virtual Organizations Constitution, in: Proceedings of Cracow Grid Workshop 2010 (CGW10), ACC Cyfronet AGH, Krakow, 2011.

[17] W. Funika, B. Kryza, R. Slota, J. Kitowski, K. Skalkowski, J. Sendor, D. Krol, Monitoring of SLA Parameters within VO for the SOA Paradigm., in: R. Wyrzykowski, J. Dongarra, K. Karczewski, J. Wasniewski (Eds.), PPAM (2), Vol. 6068 of Lecture Notes in Computer Science, Springer, 2009, pp. 115–124.

[18] J. S. Perry, Java Management Extensions, 1st Edition, O'Reilly, Beijing, 2002.

[19] M. Bubak, W. Funika, M. Smetek, Z. Kilianski, R. Wismller, Event handling in the j-ocm monitoring system., in: R. Wyrzykowski, J. Dongarra, M. Paprzycki, J. Wasniewski (Eds.), PPAM, Vol. 3019 of Lecture Notes in Computer Science, Springer, 2003, pp. 344–351.

[20] PLGrid Project website.
URL `http://www.plgrid.pl/en`

[21] A.-J. Berre, B. Elves?ter, N. Figay, C. Guglielmina, S. Johnsen, D. Karlsen, T. Knothe, S. Lippe, The athena interoperability framework., in: R. Jardim-Gonalves, J. P. Mller, K. Mertins, M. Zelm (Eds.), IESA, Springer, 2007, pp. 569–580.

[22] I. Zinnikus, C. Hahn, K. Fischer, A model-driven, agent-based approach for the integration of services into a collaborative business process., in: L. Padgham, D. C. Parkes, J. P. Mller, S. Parsons (Eds.), AAMAS (1), IFAAMAS, 2008, pp. 241–248.

[23] B. Burmeister, M. Arnold, F. Copaciu, G. Rimassa, Bdi-agents for agile goal-oriented business processes., in: M. Berger, B. Burg, S. Nishiyama (Eds.), AAMAS (Industry Track), IFAAMAS, 2008, pp. 37–44.

[24] J. Patel, W. T. L. Teacy, N. R. Jennings, M. Luck, S. Chalmers, N. Oren, T. J. Norman, A. D. Preece, P. M. D. Gray, G. Shercliff, P. J. Stockreisser, J. Shao, W. A. Gray, N. J. Fiddian, S. G. Thompson, Conoise-g: agent-based virtual organisations., in: H. Nakashima, M. P. Wellman, G. Weiss, P. Stone (Eds.), AAMAS, ACM, 2006, pp. 1459–1460.

[25] H. L. Cardoso, A. Malucelli, A. P. Rocha, E. Oliveira, Institutional services for dynamic virtual organizations, in: A. O. Luis M. Camarinha-Matos, Hamideh Afsarmanesh (Ed.), Collaborative Networks and Their Breeding Environments IFIP TC5 WG 5.5 Sixth IFIP Working Conference on VIRTUAL ENTERPRISES, 26?28 September, 2005, Valencia, Spain, 2005, pp. 521–528.